Control Breaks

This document describes how the execution of a statement can be made dependent on a control break, and how control breaks can be used for the evaluation of Natural system functions.

The following topics are covered:

Use of Control Breaks

A control break occurs when the value of a control field changes.

The execution of statements can be made dependent on a control break.

A control break can also be used for the evaluation of Natural system functions.

System functions are discussed in System Variables and System Functions. For detailed descriptions of the system functions available, refer to the Natural System Functions documentation.

AT BREAK Statement

With the statement AT BREAK, you specify the processing which is to be performed whenever a control break occurs, that is, whenever the value of a control field which you specify with the AT BREAK statement changes. As a control field, you can use a database field or a user-defined variable.

The following topics are covered below:

Control Break Based on a Database Field

The field specified as control field in an AT BREAK statement is usually a database field.

Example:

```
...
AT BREAK OF DEPT
   statements
END-BREAK
...
```


In this example, the control field is the database field DEPT; if the value of the field changes, for example, FROM "SALE01" to "SALE02", the *statements* specified in the AT BREAK statement would be executed.

Instead of an entire field, you can also use only part of a field as a control field. With the slash-*n*-slash notation "/*n*/" you can determine that only the first *n* positions of a field are to be checked for a change in value.

Example:

```
...
AT BREAK OF DEPT /4/
   statements
END-BREAK
...
```

In this example, the specified **statements** would only be executed if the value of the first 4 positions of the field DEPT changes, for example, FROM "SALE" to "TECH"; if, however, the field value changes from "SALE01" to "SALE02", this would be ignored and no AT BREAK processing performed.

Example of AT BREAK Statement using a Database Field:

```
** Example Program 'ATBREX01'
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 NAME
  2 CITY
  2 COUNTRY
  2 JOB-TITLE
  2 SALARY (1:1)
END-DEFINE
*
READ (5) MYVIEW BY CITY WHERE COUNTRY = 'USA'
  DISPLAY CITY (AL=9) NAME 'POSITION' JOB-TITLE 'SALARY' SALARY (1)
  AT BREAK OF CITY
    WRITE / OLD(CITY) (EM=X^X^X^X^X^X^X^X^X^X^X^X)
          5X 'AVERAGE:' T*SALARY AVER(SALARY(1)) //
          COUNT(SALARY(1)) 'RECORDS FOUND' /
  END-BREAK
  AT END OF DATA
    WRITE 'TOTAL (ALL RECORDS):' T*SALARY(1) TOTAL(SALARY(1))
  END-ENDDATA
END-READ
END
```

In the above program, the first WRITE statement is executed whenever the value of the field CITY changes.

In the AT BREAK statement, the system functions OLD, AVER and COUNT are evaluated (and output in the WRITE statement).

In the AT END OF DATA statement, the system function TOTAL is evaluated.

The program produces the following output:


```
Page      1                                                97-08-19  18:17:27

  CITY               NAME                    POSITION                SALARY
---------    --------------------    -------------------------    ----------

AIKEN        SENKO                   PROGRAMMER                        31500

A I K E N                 AVERAGE:                                    31500

       1 RECORDS FOUND

ALBUQUERQ    HAMMOND                 SECRETARY                         22000
ALBUQUERQ    ROLLING                 MANAGER                           34000
ALBUQUERQ    FREEMAN                 MANAGER                           34000
ALBUQUERQ    LINCOLN                 ANALYST                           41000

A L B U Q U E R Q U E     AVERAGE:                                       32750

       4 RECORDS FOUND

TOTAL (ALL RECORDS):                                                  162500
```

Control Break Based on a User-Defined Variable

A user-defined variable can also be used as control field in an AT BREAK statement.

In the following program, the user-defined variable #LOCATION is used as control field.

```
** Example Program 'ATBREX02'
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 CITY
  2 COUNTRY
  2 JOB-TITLE
  2 SALARY (1:1)
1 #LOCATION (A20)
END-DEFINE
*
READ (5) MYVIEW BY CITY WHERE COUNTRY = 'USA'
  BEFORE BREAK PROCESSING
    COMPRESS CITY 'USA' INTO #LOCATION
  END-BEFORE
  DISPLAY #LOCATION 'POSITION' JOB-TITLE 'SALARY' SALARY (1)
  AT BREAK OF #LOCATION
    SKIP 1
  END-BREAK
END-READ
END
```

The above program produces the following output:


```
Page      1                                                97-08-19  18:21:23

     #LOCATION                 POSITION                 SALARY
--------------------    -------------------------    ----------

AIKEN USA               PROGRAMMER                        31500

ALBUQUERQUE USA         SECRETARY                         22000
ALBUQUERQUE USA         MANAGER                           34000
ALBUQUERQUE USA         MANAGER                           34000
ALBUQUERQUE USA         ANALYST                           41000
```

Multiple Control Break Levels

As explained above, the notation "/*n*/" allows some portion of a field to be checked for a control break. It is possible to combine several AT BREAK statements, using an entire field as control field for one break and part of the same field as control field for another break.

In such a case, the break at the lower level (entire field) must be specified before the break at the higher level (part of field); that is, in the first AT BREAK statement the entire field must be specified as control field, and in the second one part of the field.

The following example program illustrates this, using the field DEPT as well as the first 4 positions of that field (DEPT /4/).

```
** Example Program 'ATBREX03'
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 NAME
  2 JOB-TITLE
  2 DEPT
  2 SALARY    (1:1)
  2 CURR-CODE (1:1)
END-DEFINE
READ MYVIEW BY DEPT STARTING FROM 'SALE40' ENDING AT 'TECH10'
                    WHERE SALARY(1) GT 47000 AND CURR-CODE(1) = 'USD'
  AT BREAK OF DEPT
    WRITE '*** LOWEST BREAK LEVEL ***' /
  END-BREAK
  AT BREAK OF DEPT /4/
    WRITE '*** HIGHEST BREAK LEVEL ***'
  END-BREAK
  DISPLAY DEPT NAME 'POSITION' JOB-TITLE
END-READ
END
```



```
Page      1                                                97-08-19  18:24:16

DEPARTMENT            NAME                    POSITION
   CODE
----------    --------------------    -------------------------

TECH05        HERZOG                  MANAGER
TECH05        LAWLER                  MANAGER
TECH05        MEYER                   MANAGER
*** LOWEST BREAK LEVEL ***

TECH10        DEKKER                  DBA
*** LOWEST BREAK LEVEL ***

*** HIGHEST BREAK LEVEL ***
```

In the following program, one blank line is output whenever the value of the field DEPT changes; and whenever the value in the first 4 positions of DEPT changes, a record count is carried out by evaluating the system function COUNT.

```
** Example Program 'ATBREX04'
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 DEPT
  2 REDEFINE DEPT
    3 #GENDEP (A4)
  2 NAME
  2 SALARY (1)
END-DEFINE
WRITE TITLE '** PERSONS WITH SALARY > 30000, SORTED BY DEPARTMENT **' /
LIMIT 9
READ MYVIEW BY DEPT FROM 'A' WHERE SALARY(1) > 30000
  DISPLAY 'DEPT' DEPT NAME 'SALARY' SALARY(1)
  AT BREAK OF DEPT
    SKIP 1
  END-BREAK
  AT BREAK OF DEPT /4/
    WRITE COUNT(SALARY(1)) 'RECORDS FOUND IN:' OLD(#GENDEP) /
  END-BREAK
END-READ
END
```



```
        ** PERSONS WITH SALARY > 30000, SORTED BY DEPARTMENT **

 DEPT          NAME           SALARY
------ -------------------- ----------

ADMA01 JENSEN                  180000
ADMA01 PETERSEN                105000
ADMA01 MORTENSEN               320000
ADMA01 MADSEN                  149000
ADMA01 BUHL                    642000

ADMA02 HERMANSEN               391500
ADMA02 PLOUG                   162900
ADMA02 HANSEN                  234000

       8 RECORDS FOUND IN: ADMA

COMP01 HEURTEBISE              168800

       1 RECORDS FOUND IN: COMP
```

Automatic Break Processing

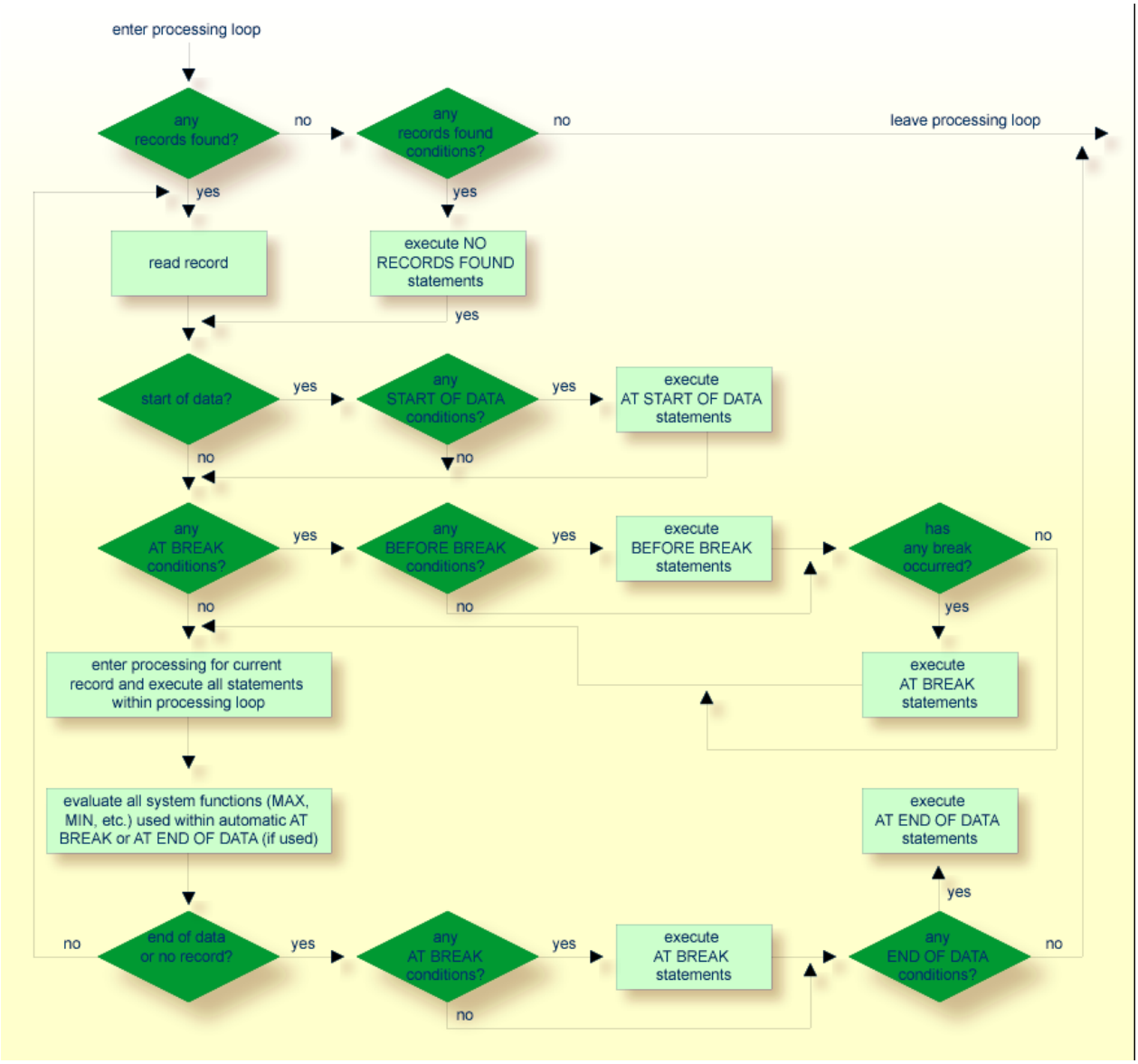
Automatic break processing is in effect for a processing loop which contains an AT BREAK statement. This applies to the following statements:

- FIND
- READ
- HISTOGRAM
- SORT
- READ WORK FILE

The value of the control field specified with the AT BREAK statement is checked only for records which satisfy the selection criteria of both the WITH clause and the WHERE clause.

Natural system functions (AVER, MAX, MIN, etc.) are evaluated for each record after all statements within the processing loop have been executed. System functions are not evaluated for any record which is rejected by WHERE criteria.

The figure below illustrates the flow logic of automatic break processing.



Example of System Functions with AT BREAK Statement

The following example shows the use of the Natural system functions OLD, MIN, AVER, MAX, SUM and COUNT in an AT BREAK statement (and of the system function TOTAL in an AT END OF DATA statement).

```
** Example Program 'ATBREX05'
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 NAME
  2 CITY
  2 SALARY    (1:1)
  2 CURR-CODE (1:1)
END-DEFINE
*
LIMIT 3
READ MYVIEW BY CITY = 'SALT LAKE CITY'
  DISPLAY NOTITLE CITY NAME 'SALARY' SALARY(1) 'CURRENCY' CURR-CODE(1)
    AT BREAK OF CITY
      WRITE /  OLD(CITY) (EM=X^X^X^X^X^X^X^X^X^X^X^X^X^X^X^X^X^X)
```

```
        31T ' - MINIMUM:' MIN(SALARY(1))  CURR-CODE(1) /
        31T ' - AVERAGE:' AVER(SALARY(1)) CURR-CODE(1) /
        31T ' - MAXIMUM:' MAX(SALARY(1))  CURR-CODE(1) /
        31T ' -     SUM:' SUM(SALARY(1))  CURR-CODE(1) /
        33T COUNT(SALARY(1)) 'RECORDS FOUND' /
   END-BREAK
   AT END OF DATA
      WRITE 22T 'TOTAL (ALL RECORDS):' T*SALARY
                TOTAL(SALARY(1))  CURR-CODE(1)
   END-ENDDATA
END-READ
END
```

```
         CITY                 NAME           SALARY   CURRENCY
-------------------- -------------------- ---------- --------

SALT LAKE CITY       ANDERSON                  50000 USD
SALT LAKE CITY       SAMUELSON                 24000 USD

S A L T   L A K E   C I T Y    - MINIMUM:      24000 USD
                               - AVERAGE:      37000 USD
                               - MAXIMUM:      50000 USD
                               -     SUM:      74000 USD
                                       2 RECORDS FOUND

SAN DIEGO            GEE                       60000 USD

S A N   D I E G O              - MINIMUM:      60000 USD
                               - AVERAGE:      60000 USD
                               - MAXIMUM:      60000 USD
                               -     SUM:      60000 USD
                                       1 RECORDS FOUND

                     TOTAL (ALL RECORDS):     134000 USD
```

BEFORE BREAK PROCESSING Statement

With the PERFORM BREAK PROCESSING statement, you can specify statements that are to be executed immediately before a control break; that is, before the value of the control field is checked, before the statements specified in the AT BREAK block are executed, and before any Natural system functions are evaluated.

Example of BEFORE BREAK PROCESSING Statement

```
** Example Program 'BEFORX01'
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 NAME
  2 FIRST-NAME
  2 SALARY (1:1)
  2 BONUS  (1:1,1:1)
1 #INCOME  (P11)
END-DEFINE
*
LIMIT 5
READ MYVIEW BY NAME FROM 'B'
```



```
  BEFORE BREAK PROCESSING
    COMPUTE #INCOME = SALARY(1) + BONUS(1,1)
  END-BEFORE
  DISPLAY NOTITLE NAME FIRST-NAME (AL=10)
          'ANNUAL/INCOME' #INCOME
          'SALARY' SALARY(1) (LC==) / '+ BONUS' BONUS(1,1) (IC=+)
  AT BREAK OF #INCOME
    WRITE T*#INCOME '-'(24)
  END-BREAK
END-READ
END
```

```
        NAME         FIRST-NAME    ANNUAL      SALARY
                                   INCOME      + BONUS
-------------------- ---------- ------------ -----------
BACHMANN             HANS             297546 =    293546
                                               +4000
                                 ------------------------
BAECKER              JOHANNES         420244 =    413644
                                               +6600
                                 ------------------------
BAECKER              KARL              52650 =     48600
                                               +4050
                                 ------------------------
BAGAZJA              MARJAN           152700 =    129700
                                               +23000
                                 ------------------------
BAILLET              PATRICK          198500 =    188000
                                               +10500
                                 ------------------------
```

User-Initiated Break Processing - PERFORM BREAK PROCESSING Statement

With automatic break processing, the statements specified in an AT BREAK block are executed whenever the value of the specified control field changes - regardless of the position of the AT BREAK statement in the processing loop.

With a PERFORM BREAK PROCESSING statement, you can perform break processing at a specified position in a processing loop: the PERFORM BREAK PROCESSING statement is executed when it is encountered in the processing flow of the program.

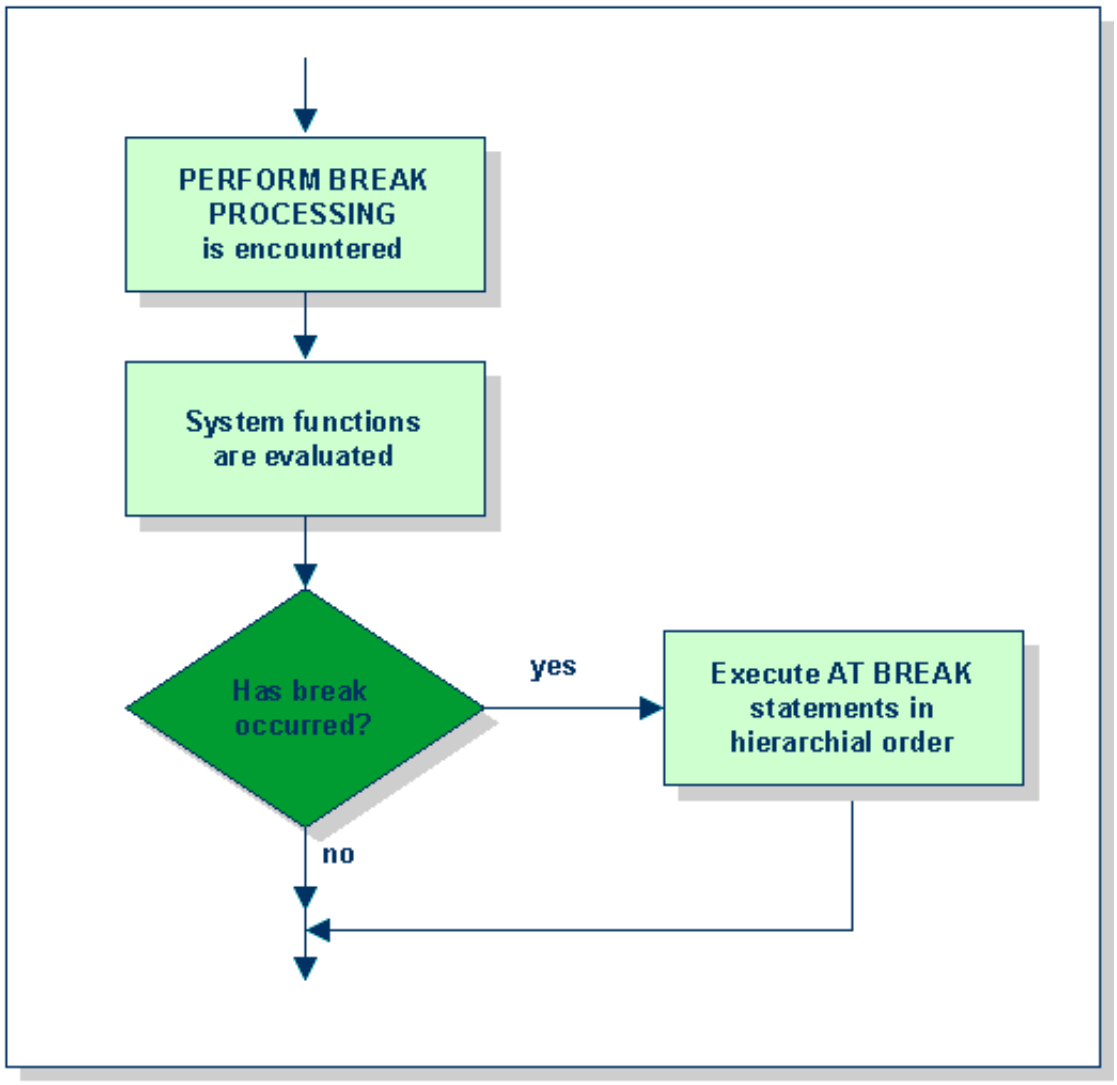
Immediately after the PERFORM BREAK PROCESSING, you specify one or more AT BREAK statement blocks:

```
...
PERFORM BREAK PROCESSING
  AT BREAK OF field1
    statements
  END-BREAK
  AT BREAK OF field2
    statements
  END-BREAK
...
```

When a PERFORM BREAK PROCESSING is executed, Natural checks if a break has occurred; that is, if the value of the specified control field has changed; and if it has, the specified statements are executed.

With PERFORM BREAK PROCESSING, system functions are evaluated *before* Natural checks if a break has occurred.

The following figure illustrates the flow logic of user-initiated break processing:



Example of PERFORM BREAK PROCESSING Statement

```
** Example Program 'PERFBX01'
DEFINE DATA LOCAL
1 MYVIEW VIEW OF EMPLOYEES
  2 NAME
  2 DEPT
  2 SALARY (1:1)
1 #CNTL    (N2)
END-DEFINE
*
LIMIT 7
READ MYVIEW BY DEPT
  AT BREAK OF DEPT              /* <- automatic break processing
    SKIP 1
    WRITE 'SUMMARY FOR ALL SALARIES         '
          'SUM:'   SUM(SALARY(1))
```



```
          'TOTAL:' TOTAL(SALARY(1))
    ADD 1 TO #CNTL
  END-BREAK
  IF SALARY (1) GREATER THAN 100000 OR BREAK #CNTL
    PERFORM BREAK PROCESSING    /* <- user-initiated break processing
      AT BREAK OF #CNTL
        WRITE 'SUMMARY FOR SALARY GREATER 100000'
              'SUM:'   SUM(SALARY(1))
              'TOTAL:' TOTAL(SALARY(1))
      END-BREAK
  END-IF
  IF SALARY (1) GREATER THAN 150000 OR BREAK #CNTL
    PERFORM BREAK PROCESSING    /* <- user-initiated break processing
      AT BREAK OF #CNTL
        WRITE 'SUMMARY FOR SALARY GREATER 150000'
              'SUM:'   SUM(SALARY(1))
              'TOTAL:' TOTAL(SALARY(1))
      END-BREAK
   END-IF
   DISPLAY NAME DEPT SALARY(1)
END-READ
END
```

```
Page      1                                                97-08-18  17:11:11

        NAME         DEPARTMENT   ANNUAL
                        CODE      SALARY
-------------------- ---------- ----------

JENSEN               ADMA01         180000
PETERSEN             ADMA01         105000
MORTENSEN            ADMA01         320000
MADSEN               ADMA01         149000
BUHL                 ADMA01         642000

SUMMARY FOR ALL SALARIES          SUM:    1396000 TOTAL:    1396000
SUMMARY FOR SALARY GREATER 100000 SUM:    1396000 TOTAL:    1396000
SUMMARY FOR SALARY GREATER 150000 SUM:    1142000 TOTAL:    1142000
HERMANSEN            ADMA02         391500
PLOUG                ADMA02         162900

SUMMARY FOR ALL SALARIES          SUM:     554400 TOTAL:    1950400
SUMMARY FOR SALARY GREATER 100000 SUM:     554400 TOTAL:    1950400
SUMMARY FOR SALARY GREATER 150000 SUM:     554400 TOTAL:    1696400
```

Further Example of AT BREAK Statement

See the following example program in library SYSEXPG:

- ATBREX06